UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

CREACIÓN DE UN ASISTENTE DE VOZ PARA MODELADO

Rubén Campos López
Tutor: Sara Pérez Soler
Ponente: Esther Guerra Sánchez

JUNIO 2020

CREACIÓN DE UN ASISTENTE DE VOZ DE MODELADO

AUTOR: Rubén Campos López
TUTOR: Sara Pérez Soler
PONENTE: Esther Guerra Sánchez

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2020

Resumen (castellano)

El procesamiento del lenguaje natural ha tenido mucha importancia en estos últimos años, dando lugar al desarrollo de bots conversacionales o chatbots que automatizan las tareas. Los agentes conversacionales se usan para automatizar tareas usando lenguaje natural y pueden ser usados por medio de voz, llamados asistentes de voz o de texto. Dentro del desarrollo software, pueden ser útiles automatizando tareas de programación y de gestión de proyectos, ayudando en todas las fases del ciclo de vida del software, en cualquier lugar, y así, reduciendo el esfuerzo empleado.

Este Trabajo Fin de Grado se centra en el desarrollo de un asistente conversacional para realizar tareas de modelado. Para ello se va a ampliar la herramienta SOCIO, un chatbot para modelado colaborativo que funciona en Telegram y Twitter, con una API REST que permite realizar las acciones de SOCIO y recibir su respuesta.

Se ha usado la herramienta Dialogflow, que permite que Google pueda interpretar el lenguaje natural, y usando técnicas de aprendizaje automático, detecta la intención del hablante. Dialogflow enviará la intención del usuario a una API intermedia, que se encarga de interpretar la intención del usuario y enviar peticiones a la API de SOCIO para que le devuelva el resultado de la acción solicitada.

Según la respuesta de SOCIO, nuestra API elaborará una respuesta al emisor, que además de dar una respuesta hablada o escrita, reflejará los resultados, si procede, a una página web creada por nuestra API, única por sesión de usuario.

El idioma usado para este asistente de voz es el Inglés, pero está abierto para un posible asistente multilenguaje, añadiendo otros idiomas como el Español, Francés, Chino, Alemán,etc.

Abstract (English)

Natural language processing has been very important in recent years, leading to the development of conversational bots or chatbots that automate tasks. Conversational agents are used to automate tasks using natural language and can be used by voice, called voice or text assistants. Within software development, they can be useful in automating programming and project management tasks, helping in all phases of the software life cycle, anywhere, and thus, reducing the effort used.

This Bachelor Thesis focuses on the development of a conversational assistant to carry out modelling tasks. For this, the SOCIO tool will be expanded, a chatbot for collaborative modelling that works on Telegram and Twitter, with a REST API that allows it to perform its actions and receive the answer.

The Dialogflow tool has been used, which allows Google to interpret natural language, and using machine learning techniques, it detects the intention of the speaker. Dialogflow will send the user's intention to an intermediate API, which is responsible for interpreting the user's intention and sending requests to the SOCIO API to return the result of the requested action.

According to SOCIO response, our API will prepare a response to the issuer, which in addition to giving a spoken or written response, will reflect the results, if applicable, to a web page created by our API, unique per user session.

The language used for this voice assistant is English, but it is open for a possible multilanguage assistant, adding other languages such as Spanish, French, Chinese, German, etc.

Palabras clave (castellano)

Chatbot, Asistente de voz, Modelado Colaborativo

Keywords (inglés)

Chatbot, Voice assistant, Collaborative Modelling

Agradecimientos

Agradezco mucho a mi tutora Sara Pérez, que me ha ayudado en todo momento no sólo a integrarme en el mundo de SOCIO, sino también en todas las dudas acerca del TFG.

También quiero agradecer a mi empresa de prácticas externas AFI, que me introdujeron en todo el tema de los bots de Google Home en su momento.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

En este apartado se expone la motivación (Sección 1.1) y los objetivos del TFG (Sección 1.2), así como la estructura de la memoria (Sección 1.3)

1.1 Motivación

En los últimos años, los asistentes de voz se han vuelto más populares y su uso es cada vez más frecuente. El procesamiento del lenguaje natural ha tenido mucho avance y a raíz de ese avance ha aumentado el uso de agentes conversacionales o chatbots. Estos bots ha ayudado en muchas tareas a los usuarios y especialmente en situaciones difíciles que normalmente generan estrés[1], y se usan como apoyo emocional[2].

Los agentes conversacionales pueden usarse mediante el texto escrito, por lo que suelen estar integrados en redes sociales, repositorios o páginas web; o mediante la voz, integrándolos en los dispositivos inteligentes, ya sean smartphones, relojes o altavoces inteligentes.

Empresas como Google, Apple o Amazon han lanzado sus propios asistentes de voz: Google Assistant, Siri[1] y Alexa[2]; integrándolos en los dispositivos móviles inteligentes, así como los altavoces inteligentes.

Varias empresas ya tienen su propio bot conversacional, y esto permite ampliar el número de personas que lo usan y que no pueden hacerlo por la vía escrita y visual. Además, los usuarios pueden realizar esas tareas mediante el asistente mientras se está haciendo otras simultáneamente. Esto reduce mucho el tiempo de trabajo y esfuerzo de la persona.

Además del tiempo reducido, estos asistentes de voz han ayudado especialmente a las personas con discapacidad, ya que facilitan la accesibilidad.

SOCIO[3], es un agente conversacional para modelado colaborativo. Su función es interpretar el lenguaje natural para crear modelos de dominio (se trata de diagramas de clase sin métodos). Funciona de forma textual, y está integrado en redes sociales como Telegram y Twitter, pudiendo ser ampliado a más redes sociales gracias al servicio REST de SOCIO.

1.2 Objetivos

En este trabajo, el objetivo principal es ampliar SOCIO para integrarlo con el asistente de voz de Google, Google Assistant, permitiendo que se interactúe con SOCIO por medio de la voz. La funcionalidad no cambia, no obstante, en vez de comandos, se usará el lenguaje natural. (Para saber más sobre SOCIO, véase la Sección 3.1).

Para alcanzar ese objetivo, se ha realizado la solución siguiendo los siguientes objetivos concretos propuestos:

[1] https://www.apple.com/es/siri/

[2] https://developer.amazon.com/es-ES/alexa

- Creación del chatbot a partir de Dialogflow que permita recoger la información del usuario y su mensaje (Para saber más sobre el framework utilizado, véase la Sección 3.2).
- Creación de un middleware que se ocupe de utilizar la información sacada de Dialogflow para enviar las peticiones adecuadas a SOCIO
- Dotar a cada usuario de SOCIO una página web que permita ver los resultados de SOCIO.

1.3 Organización de la memoria

El resto de la memoria se organiza de la siguiente manera:

- Sección 2: Estado del arte. Muestra la situación actual de los bots de desarrollo, de qué manera ayudan a los desarrolladores, herramientas de modelado, además de SOCIO.
- Sección 3: Conocimientos previos. Explica el funcionamiento de SOCIO y las ventajas de Dialogflow y Google Assistant.
- Sección 4: Diseño. Detalla cómo se ha diseñado la solución y su funcionamiento.
- Sección 5: Desarrollo. Detalla el desarrollo de la herramienta diseñada, su arquitectura, las bases de datos y las páginas web.
- Sección 6: Integración, pruebas y resultados. Se muestra una prueba de todas las acciones posibles en Google Assistant.
- Sección 7: Conclusiones y trabajo futuro.

2 Estado del arte

En este apartado se numeran ejemplos de bots de desarrollo (Sección 2.1) y herramientas de modelado (Sección 2.2).

2.1 Bot de desarrollo

Los bots para ayudar a los desarrolladores son cada vez más avanzados y reducen mucho el trabajo empleado. Hoy en día, destacan los bots o chatbots que asisten a los desarrolladores en todas las etapas del ciclo de vida de software, sobre todo de gestión y mantenimiento. En este apartado se muestran algunos ejemplos sacados de la referencia [4] (todos Open Source):

- First timers[5]: como su nombre indica, es un bot para integrar nuevos usuarios. Está hecho a partir de Probot, que es una herramienta para hacer aplicaciones en Github. Su objetivo no es solo guiar a los nuevos usuarios cuando hacen su primer Commit, sino familiarizarse con Github con opciones como Pull Request, que normalmente un nuevo usuario no lo maneja con facilidad.
- Git Enforcer[6]: este bot tiene varios usos, pero principalmente se dedica a que los issues o pull requests mantengan las reglas personalizadas hechas por los managers del proyecto, de tal manera que se puedan corregir para que dichas reglas se cumplan. Se usa en Github.
- CSSRooster[7]: se trata de un bot que se ocupa de una sola cosa: nombrar clases CSS a partir de un documento HTML. Coge todos los estilos en línea de HTML, y a partir de dónde están esos estilos (por ejemplo, en un listado o en un párrafo) los nombra y los guarda en un archivo CSS. Disponible en Huula[3].
- DependaBot[8]: este bot detecta cualquier actualización en las dependencias de tu código, por ejemplo, en proyectos con Gradle o Maven, y crea Pull Requests si alguna dependencia de tu código está desactualizada. El usuario únicamente debe comprobar que todo esté correctamente a su criterio, y realizar un Merge. Este bot además está optimizado para que no surjan conflictos. Disponible para Github, Ruby together[4], Python package index[5] y Mozilla[6].
- Sedy[9]: es un bot que cuenta como un usuario más de Github, realiza pull request con correcciones ortográficas haciendo comandos sed de Linux, de ahí el nombre. Disponible para repositorios como Github y sus derivados.
- AWS chatbot[10]: este bot, a diferencia de los anteriores, no actúa en repositorios Github, sino en la red social Slack[7]. Permite interactuar y monitorizar con los servicios web de Ámazon, así como recibir alertas.

[3] https://huu.la/

[4] https://rubytogether.org/

[5] https://pypi.org/

[6] https://www.mozilla.org/es-ES/

[7] https://slack.com/intl/es-es/

- Hubot[11]: es un bot que puede interactuarse por Slack. Se trata de un chat de código abierto que permite programar en CoffeeScript[8].

Estos bots son empleados especialmente en el propio código y en repositorios, y utiliza el texto escrito. Para en el caso de las redes sociales, los bots de desarrollo están presentes en Slack principalmente.

2.2 Herramientas de modelado

Las herramientas de modelado son fundamentales para todas las fases del ciclo de vida del software. Su objetivo es ayudar a los desarrolladores a realizarlo de manera más cómoda posible, dotando de muchas funcionalidades para cada tipo de diagrama en el diseño. La mayoría se basan en el lenguaje estándar UML[12]; sin embargo, existen varias herramientas que no se basan en ese lenguaje. A continuación, mostramos algunas de ellas:

- ArgoUML[13]: es una herramienta offline hecho en java, una de las que más años tiene. Contiene muchos tipos de diagramas y ha sido la aplicación estándar durante muchos años. Sin embargo, su última versión tiene más de 8 años de antigüedad, por lo que ya no está en mantenimiento.
- Gliffy[14]: es una herramienta online que soporta todo tipo de diagramas UML, y además la arquitectura AWS (Amazon Web Services) y ERD (Modelo Entidad-Relación). Se puede integrar con Atlassian. Se pueden importar o exportar proyectos. Es colaborativo, por lo que varios usuarios pueden editar y otorgar distintos permisos a cada uno.
- Draw.io[15]: una herramienta muy similar a Gliffy, con una versión online gratuita con la que se puede compartir por OneDrive o GoogleDrive. A diferencia del anterior, Draw.io es menos colaborativo que Gliffy, ya que depende de compartir su archivo con usuarios de Google Drive. Sin embargo, permite hacer unos cambios más precisos estéticamente.
- Cacoo[16]: posee también muchos diagramas de todo tipo, además de los mencionados anteriormente, como diagramas de Azure, diagramas de Gantt, diagramas de flujo, e incluso plantillas para diseño de aplicaciones móviles. Lo que más destaca de esta herramienta es su sistema colaborativo, ya que posee muchas herramientas para trabajar en equipo, como es mensajería, comentarios y modificar en el mismo proyecto en tiempo real.
- Cawemo[17]: esta herramienta está especializada en BPMN[18] (Bussiness Process Model and Notation). Es completamente gratuita, y aunque su contenido es limitado por la notación, posee muchas herramientas colaborativas, además de exportación en XML.

Todas las herramientas mencionadas anteriormente se editan mediante interfaz gráfica, utilizando el arrastrar del ratón para coger cada elemento e insertarlo en el diagrama.

[8] https://coffeescript.org/#overview

3 Conocimientos Previos

En este apartado se explicará el funcionamiento de la herramienta SOCIO (Sección 3.1), Dialogflow (Sección 3.2) y el asistente de voz Google Assistant (Sección 3.3).

3.1 SOCIO

SOCIO[19] es un asistente conversacional de modelado, que crea diagramas de dominio mediante mensajes en lenguaje natural. Está integrado en redes sociales, actualmente en Telegram y Twitter, pudiendo añadirse a más redes (en nuestro caso añadimos Google Assistant).

3.1.1 Funcionamiento

SOCIO permite las siguientes acciones:

- Gestión de proyectos:
  - Crear un proyecto: cada modelo está asociado a un proyecto, por lo que se debe crear uno antes de realizar tareas de modelado. SOCIO permite crear proyectos con un nombre dado por el usuario y una visibilidad (pública, protegida o privada) respecto a los demás usuarios. El usuario que crea el proyecto pasa a ser el administrador de este, lo que le permite las siguientes acciones de gestión de proyectos. Los proyectos se guardan mediante la forma "Canal/administrador/nombre". Canal es la red social origen del proyecto, administrador es el usuario que creó el proyecto y nombre es el nombre del proyecto.
  - Dar y eliminar permisos: el administrador puede añadir usuarios a su proyecto con permisos de lectura y/o escritura. Estos usuarios pueden ser de otra red social distinta al del administrador. Esos permisos se pueden cambiar o incluso eliminar.
  - Cambiar la visibilidad: SOCIO permite cambiar la visibilidad del proyecto, restringiendo o no, la entrada a los usuarios.
  - Eliminar proyectos: si es el administrador del proyecto, tiene la opción de eliminar el proyecto, borrando todo su contenido.
- Escritura de modelos:
  - Realizar acciones de modelado: La acción principal de SOCIO es realizar tareas de modelado mediante el lenguaje natural, respondiendo con la creación o actualización del diagrama de dominio correspondiente al proyecto asignado (Véase figura 3.1).
  - Deshacer: SOCIO permite a los usuarios deshacer las últimas acciones de modelado, volviendo al estado anterior.
  - Rehacer: si hay alguna acción de modelado previamente deshecha, SOCIO permite que dichas acciones puedan rehacerse.
- Lectura de proyectos:
  - Obtener estadísticas: todas las acciones y mensajes hechos por los usuarios generan tablas con las estadísticas, transformándolas en gráficas. SOCIO permite ver las estadísticas de las acciones, mensajes y de porcentaje de autoría, pudiendo especificar por un usuario en concreto en los dos primeros.

- Historial de mensajes: SOCIO guarda los mensajes en un historial que el usuario puede visualizar.
- Validar el modelo: Para dar más flexibilidad con el uso del lenguaje natural, SOCIO permite realizar acciones de modelado que dejen el modelo en estados no válidos, por ejemplo, con atributos sin tipar. Por ello, tiene una acción para validar el modelo que notifica los posibles errores que haya, permitiendo al usuario corregirlos.
- Obtener el modelo: SOCIO permite obtener en un fichero el modelo construido, en un fichero de formato ecore, que puede abrirse mediante Eclipse[9].

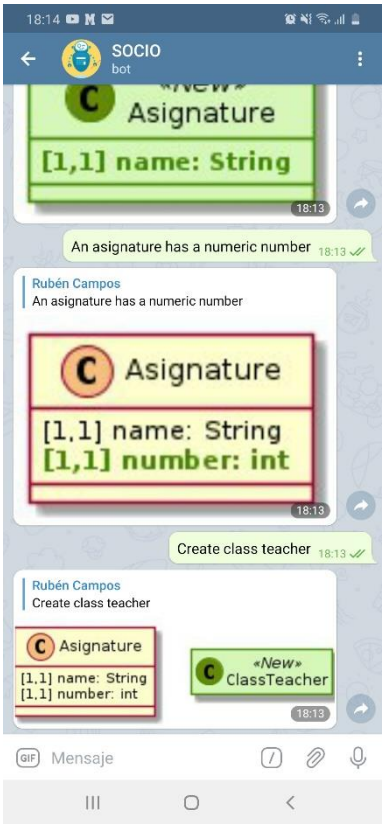


Figura 3-1: Ejemplo de uso de SOCIO en Telegram

SOCIO posee una API REST que permite realizar peticiones para ejecutar dichas acciones. La API recibe objetos en formato JSON y responde según la petición las imágenes de los modelos y estadísticas, la información de los proyectos o el fichero ecore del modelo.

[9] https://www.eclipse.org/

3.2 Dialogflow

Dialogflow[10] es una herramienta para crear chatbots. Se trata de un servicio de Google que procesa el lenguaje natural y puede fabricar una respuesta por si misma o a partir de un mensaje enviado por la API conectada a ese bot.

3.2.1 Funcionamiento

Dialogflow identifica qué es lo que quiere el usuario a través de elementos llamados "Intenciones". Cada intención tiene asociadas unas frases concretas que el desarrollador los introduce, de manera que, a partir de esas frases, Dialogflow entrena el bot usando técnicas de aprendizaje automático.

La figura 3-2 muestra un ejemplo de una intención llamada Help. Contiene tres frases de entrenamiento con las que el usuario utilizará para activar esa intención.

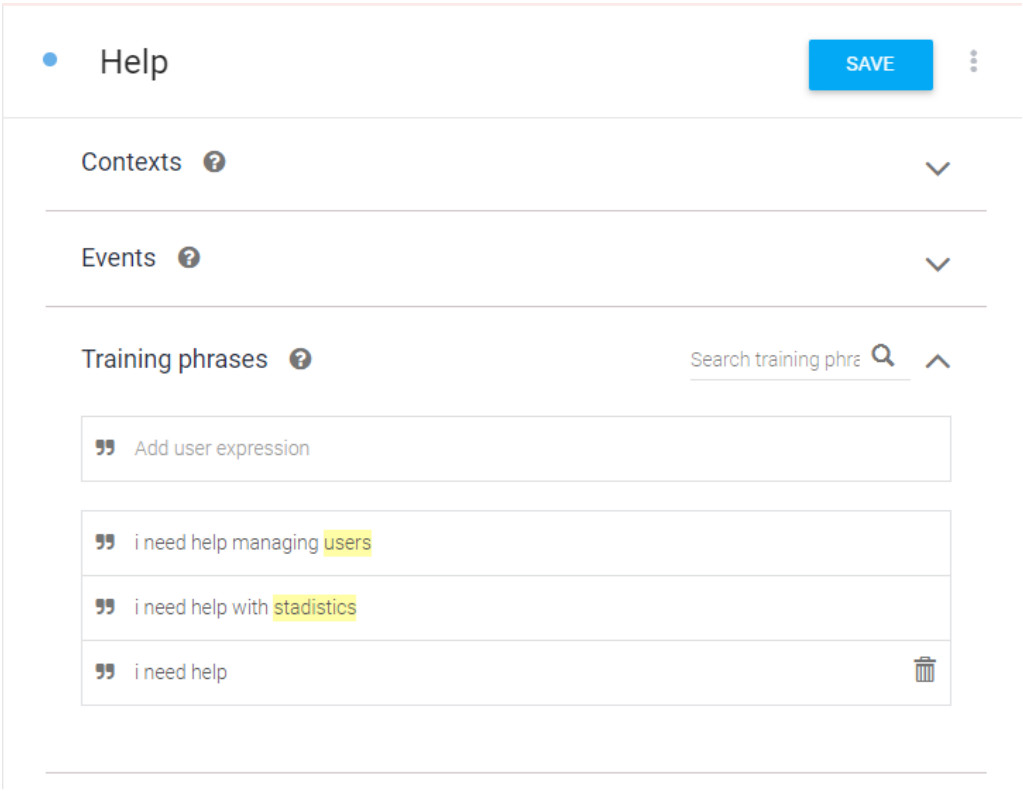


Figura 3-2: Contenido de una intención

Las intenciones pueden definir parámetros, piezas de información que el chatbot recolecta de las interacciones de los usuarios, o que debe preguntar por ellas. Los parámetros tienen un tipo asociado el cual se llama entidad. Hay entidades predefinidas como números, textos o fechas, y se pueden definir nuevas entidades como listas de palabras y sus sinónimos.

[10] https://dialogflow.com/

La figura 3-3 muestra la entidad redo-command, que contiene la palabra redo con sus sinónimos.

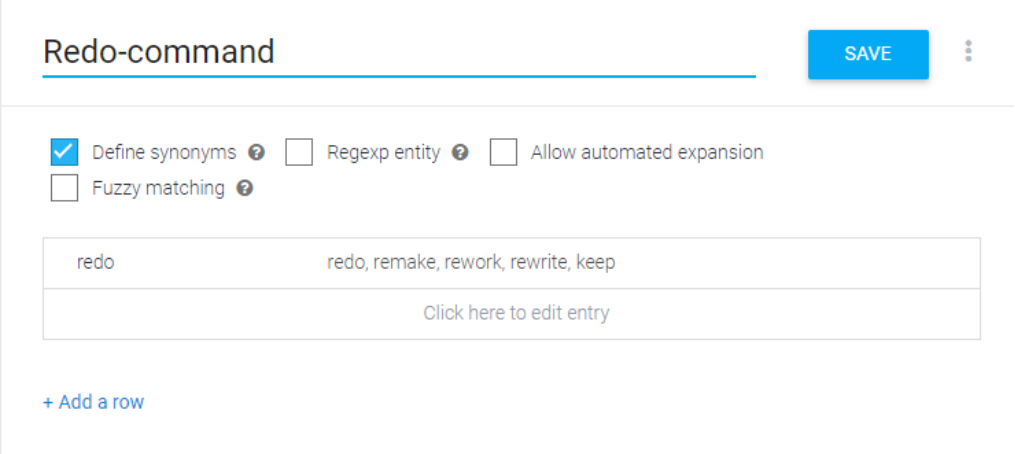


Figura 3-3: Contenido de una entidad

Hay algunas veces que una misma frase puede ser utilizada en intenciones diferentes. Depende del "Contexto". Dialogflow permite crear contextos tras realizar una acción determinada. Los contextos se utilizan para definir la estructura de la conversación. De esta manera algunas intenciones solo son accesibles tras la activación de determinados contextos en intenciones previas. También sirve para almacenar cierta información si se necesita. Los contextos tienen un tiempo límite que se puede configurar, pero en ningún caso llegan a ser ilimitados.

Dialogflow permite realizar llamadas a servicios externos. En estas llamadas se manda la información de la intención que ha sido seleccionada, los contextos, así como la información almacenada en los parámetros. El servicio debe disponer de una API publica, utilizar certificado SSL y además debe responder en un tiempo no inferior a 15 segundos, de lo contrario, se cierra el chatbot. Todo esto se puede ver en el apartado de "Fulfillment". En la figura 3-4 se muestra la ventana de fulfillment, con la url de nuestra API. Las cabeceras y el usuario son opcionales, que sirven para entregar información adicional del usuario y otros datos en cada llamada.

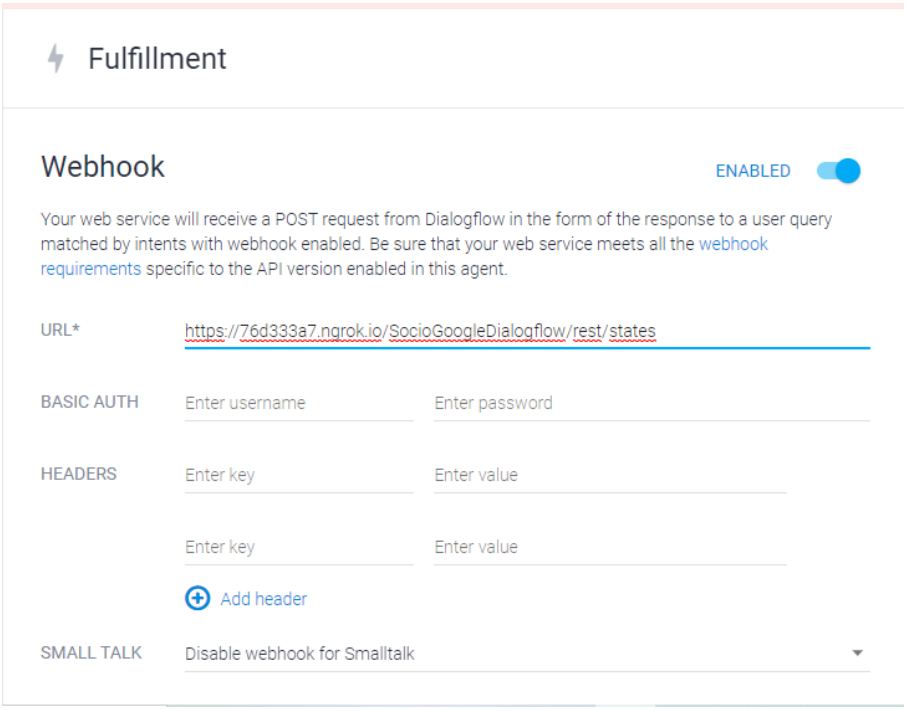


Figura 3-4: Contenido del fullfilment

3.2.2 Ventajas

Las razones por las que este framework es el más indicado de usar para nuestro TFG son las siguientes:

- Optimizado para Google Assistant: Casi todos los dispositivos Android poseen Google Assistant, y a partir de esos dispositivos se ha basado esta plataforma. En Dialogflow permite además que los dispositivos que estén vinculados con el correo puedan utilizarse para testearse en esos dispositivos y comprobar su funcionamiento.
- Multilenguaje: Un bot puede gestionar más de un idioma en sus acciones. Esta ventaja hace innecesario crear un segundo bot igual que el primero sólo porque el idioma sea distinto.
- Soporta más redes: Aunque está más dirigido a crear chatbots para GoogleAssitant, los chatbots de Dialogflow también se pueden utilizar en redes sociales como Telegram, Skype, WhatsApp, Slack, … Incluso para Amazon Alexa.
- Permite API inline: también dentro de Dialogflow se puede hacer una API inline en lenguaje Node.js. Tanto la API como el chatbot pertenecen al mismo proyecto Google Cloud, y pueden enlazarse a un proyecto de Firebase.
- Precio: aunque tenga funciones Premium, todo lo imprescindible que se necesita para crear el chatbot es completamente gratuito.

3.3 Google Assistant

Google Assistant[11] es un asistente de voz disponible en dispositivos Android e IOS (en Android viene preinstalado en algunos dispositivos).

3.3.1 Funcionamiento

Para acceder a Google Assistant:

- Si está preinstalado, se puede acceder a él con la frase "OK, Google" o accediendo al icono de la aplicación.
- Si no lo está, se deberá instalar la aplicación a través de App Store o Play Store

Una vez conectado con el asistente, se inicia sesión con el correo electrónico de Google. Cuando esté iniciado, puede enviar una frase cualquiera, ya sea hablado o escrito.

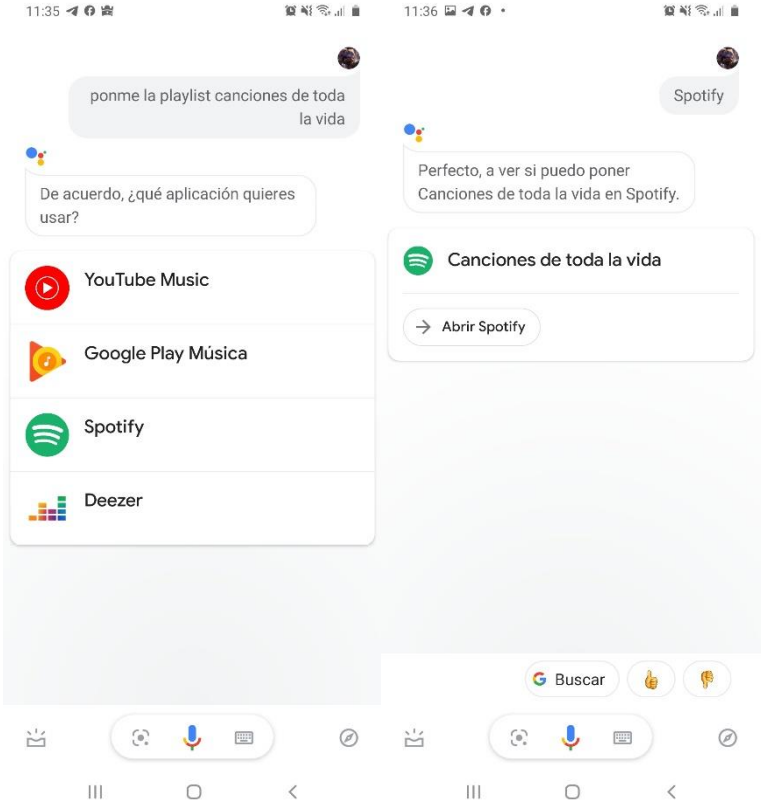


Figura 3-5: Ejemplo de uso de Google Assistant

3.3.2 Ventajas

Google Assistant ofrece las siguientes ventajas, además de estar preinstalado en Android:

[11] https://assistant.google.com/

- Multiplataforma: está disponible para IOS y Android, además de poder usarse en muchos dispositivos, no sólo en móviles, también en relojes, altavoces, televisores y pantallas inteligentes
- Optimizado para Google y otras aplicaciones: tiene interacciones más avanzadas para servicios de Google y otras apps como Spotify, Netflix, etc.
- Mayor capacidad de respuesta: Google Assistant puede responder preguntas mucho más complejas.[20]

4 Diseño

En este apartado se explica el funcionamiento del TFG (Sección 4.1)

4.1 Funcionamiento

El funcionamiento de la solución es similar al de SOCIO, como se ha explicado en la sección 3.1.1; sin embargo, debido a que se va a emplear el lenguaje natural y se utilizará en altavoces inteligentes, contiene los siguientes elementos:

- Inicio de sesión: los usuarios que se conecten a SOCIO tienen que iniciar sesión con su cuenta de Google. En caso de ser la primera vez que se conecte al bot, Google pedirá aceptar los términos y condiciones. Esto es obligatorio aceptarlo, porque de lo contrario, no se podrá conectar al bot. Si el usuario usa dos dispositivos en una misma cuenta, tendrá que aceptar los términos en los dos.
- SetProject: es una acción que consiste en entrar en un proyecto existente que el usuario puede acceder (tiene que ser el administrador o tener permisos). Esta acción es necesaria hacerla para poder realizar el resto de las acciones. Cuando se crea un proyecto, se hace un setProject automáticamente del mismo proyecto.
- Acciones que no necesitan setProject: las acciones removeProject(que borra el proyecto que indica el usuario), Bye(la acción de salida del bot) y help(la acción para pedir ayuda sobre las acciones posibles) pueden utilizarse en cualquier momento sin necesidad de entrar en un proyecto.
- Página Web: debido a que los altavoces inteligentes no pueden responder de manera visual, los diagramas construidos de SOCIO se reflejan en una página web única para cada usuario. Además del último resultado, se muestran los dos anteriores si lo hubiera, cualquier otro dato como son las estadísticas del proyecto o el historial de mensajes, según lo que solicite el usuario. Dicha página se actualizará cada varios segundos. Para acceder a esa página web, el usuario debe pedir la URL al bot. En caso de utilizar en móvil Google Assistant, se envía un enlace de la página web (Véase Sección 6.2). Cuando se hace un SetProject, la página web cambiará su contenido al nuevo proyecto.
- Obtener el proyecto: para obtener el proyecto, tras realizar la petición, el bot le responderá con una URL donde estará ubicado el archivo. Dicha URL es la misma para todos los usuarios.

En la figura 4-1 se muestra el diagrama de estados de la solución que resume el inicio de sesión y las acciones posibles.

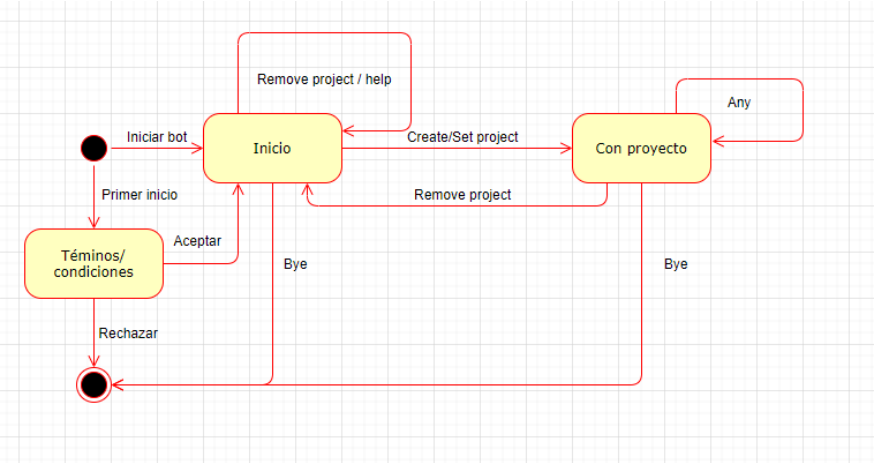


Figura 4-1: Diagrama de estados de las acciones de la solución

5 Desarrollo

En este apartado se explica el desarrollo de esta herramienta. Principalmente se explica la arquitectura (Sección 5.1), el desarrollo y las APIs usadas en el backend (Sección 5.2), la gestión de las páginas web (Sección 5.3) yel contenido de la base de datos (Sección 5.4).

5.1 Arquitectura

En este trabajo, para ampliar el alcance de SOCIO a los altavoces inteligentes, es necesario crear una nueva solución, ya que se va a emplear el lenguaje natural para interactuar con SOCIO. Por ese motivo, tenemos que hacer que el usuario use el lenguaje natural con el asistente de voz sin necesidad de modificar el funcionamiento de SOCIO.

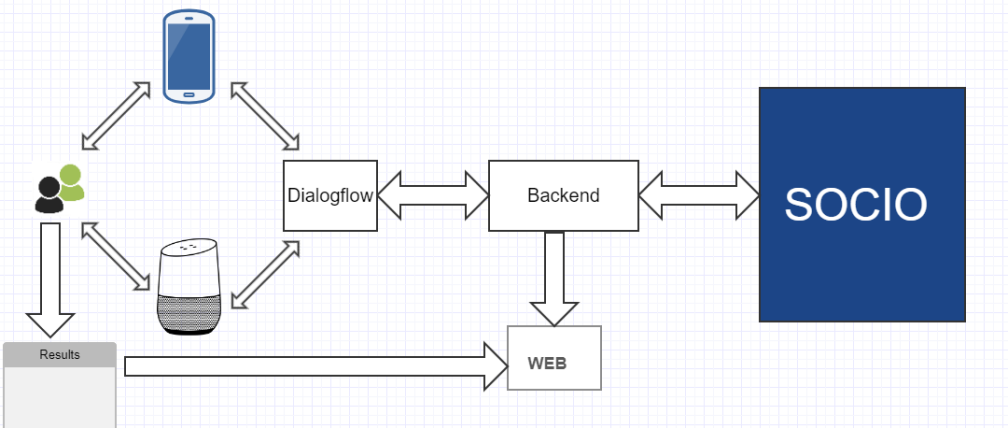


Figura 5-1: Arquitectura del bot

El proceso de enviar una solicitud al bot y obtener la respuesta es el siguiente:

El usuario, mediante cualquier dispositivo que contenga Google Assistant habla con el asistente para cualquier acción dirigida a SOCIO, ya sea hablado o escrito.

Dicho mensaje lo analiza Dialogflow. Cada acción diferente de SOCIO se corresponde con una intención. En el caso de hacer una frase cualquiera que no corresponde a ninguna intención especial, corresponde a la acción de "do" de SOCIO. A esto se llama ***Default Fallback Intent***. Según el contexto y la intención analizada, envía esa información al backend.

Debido a que se necesita una URL pública para nuestro backend en Dialogflow, nuestro servidor local se traduce a una URL producida por "ngrok". De esta manera, se consigue conectar Dialogflow a nuestro backend.

El backend clasifica toda la información del usuario y del mensaje, y comprueba si hay algún proyecto seleccionado. A continuación, en función de la intención recibida, realiza una petición a SOCIO con la información en formato JSON.

Para poder mostrar los resultados al usuario, una vez de obtener la respuesta de SOCIO, el backend crea o actualiza la página web asignada al usuario con toda la información de esa respuesta. Tras las actualizaciones, se transmite la respuesta a Dialogflow, para que éste le envíe la respuesta al usuario.

5.2 Desarrollo del backend

Siguiendo el esquema de la sección 5.1, el backend para gestionar los mensajes de Google Assistant está basado en la arquitectura REST e implementado en JAVA. Para utilizarlo como servicio REST se ha usado la API Jersey[12]. Jersey nos permite que todo POST del bot se dirija a nuestro método de control.

Para identificar al usuario, cada uno tiene su id de Google mediante un token de **JWT**[13] **(JSON Web Tokens)**, que se trata de un token codificado que contiene toda la información del usuario de Google, y es diferente para cada usuario y para cada dispositivo móvil. Si no se manda id nada más iniciar sesión, Google pedirá al usuario utilizar la sesión del usuario para el bot. Si acepta los términos y condiciones de Google Assistant al usar este chatbot, se le asigna un id. Ese id pasa a la tabla de gestión de usuarios de la base de datos (Véase 5.3), donde, si existe su correo de Google en la tabla, le devolverá el id de SOCIO y su id de sesión correspondiente. Éste último servirá para acceder a su propia página web. En caso de que no exista, se le creará uno nuevo con un nuevo id de SOCIO y una sesión para las páginas web.

Como sólo se recibe información para enviarla a SOCIO y responder al emisor con la respuesta recibida de SOCIO, nuestro backend sólo contiene la clase **States**, que es la encargada de hacer todas las tareas de gestión de los mensajes y de crear las correspondientes páginas web por usuario.

El método ***printJson*** es el principal y el que nuestro bot manda el mensaje. A partir del JSON recibido, se recoge toda la información esencial: el mensaje del usuario, los datos del usuario, la intención detectada y el contexto, si existe, que nos indica el proyecto en el que está modificando el usuario. Recogida toda esa información, se envía la petición correspondiente a SOCIO.

Tras recibir la respuesta de SOCIO, el método ***printJSON*** llama a ***setResponse*** cuyo objetivo es construir el JSON de respuesta según la información obtenida de SOCIO.

SetResponse actualiza el contexto de Dialogflow, que contiene el proyecto con el que está iniciado la sesión, lo hacen mediante la forma “Canal-Nick-Nombre”, donde Canal es la red social origen del proyecto, Nick es el apodo del administrador del proyecto y Nombre el nombre de éste.

Dentro de la clase **States**, existe una clase privada llamada **UpdateThread**, que extiende la clase **Thread** de Java, que se trata de un hilo que ha de ser único, cuya función consiste en actualizar todas las páginas web involucradas en el proyecto modificado tanto por el

[12] https://github.com/eclipse-ee4j/jersey

[13] https://jwt.io/

usuario que escribe el mensaje como por otros usuarios que lo hacen desde otro canal. El método ***start*** se registra en SOCIO en el canal correspondiente, que nuestro caso es Google, para poder recibir las actualizaciones de los proyectos que pertenezcan a ese canal. El método ***run***, utiliza un bucle infinito para estar atento a las actualizaciones en todo momento. Cuando detecta la actualización, actualiza los mensajes guardados y los archivos multimedia, y si algunos de esos proyectos están presentes en una de las páginas web, actualiza dicha página.

PrintJSON crea un **UpdateThread** cuando un usuario inicia sesión, pero antes llama al método ***getUpdateThread***, para comprobar si ya existe un hilo de ese tipo. En caso de que ya exista uno, se omite la creación de otro nuevo hilo.

5.3 Páginas web

Para crear las páginas web, se han tomado dos enfoques a la hora de gestionarlos y mostrarlos: una página web por proyecto o una página web por usuario. Se decidió el segundo por las siguientes razones:

- Soporte a cambio de proyecto: si un usuario quiere cambiar de proyecto, con tan sólo una frase, la misma página se le cambia todo el contenido, en vez de buscar una nueva URL para acceder a ella.
- Soporte a proyectos de otros canales: si un usuario de Telegram o Twitter da permiso a un usuario de Google en uno de sus proyectos de SOCIO, dicho proyecto puede ser accesible y, por tanto, mostrarse en la página. Además, reafirma la razón anterior, ya que no hará falta crear una nueva página web por un proyecto ya creado en otro canal.

En el momento de crear las páginas web, toda la información recibida de SOCIO, siempre que sea válida, se le manda al método ***sendResult***, que cambia respecto a la información que se le envía, ya sea una imagen o texto.

En caso de que se vaya a borrar un proyecto, se llama al método ***resetJsp***, que vacía todo el contenido de la página.

Para los archivos multimedia que utilizan las páginas web, se almacenarán:

- 1 archivo de texto para los mensajes por cada proyecto.
- 3 imágenes con los 3 últimos resultados de modificaciones del proyecto y una imagen por cada tipo de estadísticas del proyecto.
- 3 archivos de texto para las explicaciones de ayuda cuando el usuario lo solicite.

Para crear contenido HTML se ha usado la API Jsoup[14], que nos ayuda a crear el contenido HTML que se necesita, añadiendo y modificando clases, id y etiquetas sin tener que hacer muchas comprobaciones de Strings.

Las páginas web generadas para este proyecto son JSP, que de momento tienen sólo contenido HTML, pero para el futuro, estas páginas podrían llevar contenido dinámico por

[14] https://jsoup.org/

clases Java. Dentro de las páginas, para dotarlas con mejor aspecto, se ha utilizado Jquery[15] para cambios en la página, y Bootstrap[16] para el estilo de la página tanto para ordenador como para móviles.

5.4 Base de datos

Nuestro backend estará conectado a una base de datos local, cuya función consiste en almacenar los datos esenciales del usuario y traducir el id de Google al id de SOCIO, debido a que el primero es un String numérico que contiene 24 caracteres, que excede el máximo del id de SOCIO que es Long. Además, también genera el id de sesión que será el que se usará para identificar a las páginas web del usuario. Contiene los siguientes elementos:

- Id de Google (String): id que nos envía Google
- Id de Socio (Long): id para SOCIO que se genera aleatoriamente. Único.
- Id de sesión (Long): id de las páginas web del usuario. Único
- Nick (String): el nick que se usa en SOCIO para iniciar sesión.

Toda la solución del backend se lanza en un servidor local. Ese servidor está configurado de manera que cualquier envío al servidor, se transmita a nuestra clase asociada.

Se ha utilizado una base de datos local de Postgresql[17], utilizando conexión JDBC para poder acceder a ella desde nuestro backend.

[15] https://jquery.com/

[16] https://getbootstrap.com/

[17] https://www.postgresql.org/

6 Integración, pruebas y resultados

En este apartado se explica la interfaz de Google Assistant y de la página web (Sección 6.1) y las pruebas del TFG realizado (Sección 6.2).

6.1 Integración

La interfaz de nuestro bot en Google Assistant se muestra de forma similar que en Telegram (figura 3-5), con la diferencia de que al invocar el bot, se abre una nueva sección de diálogo de título Modelling bot SOCIO. En esta sección, todas las frases que se digan o escriban se enviarán al backend. En la figura 6-1 se muestra la invocación del bot y la vista de Google Assistant antes y después de invocar al bot:

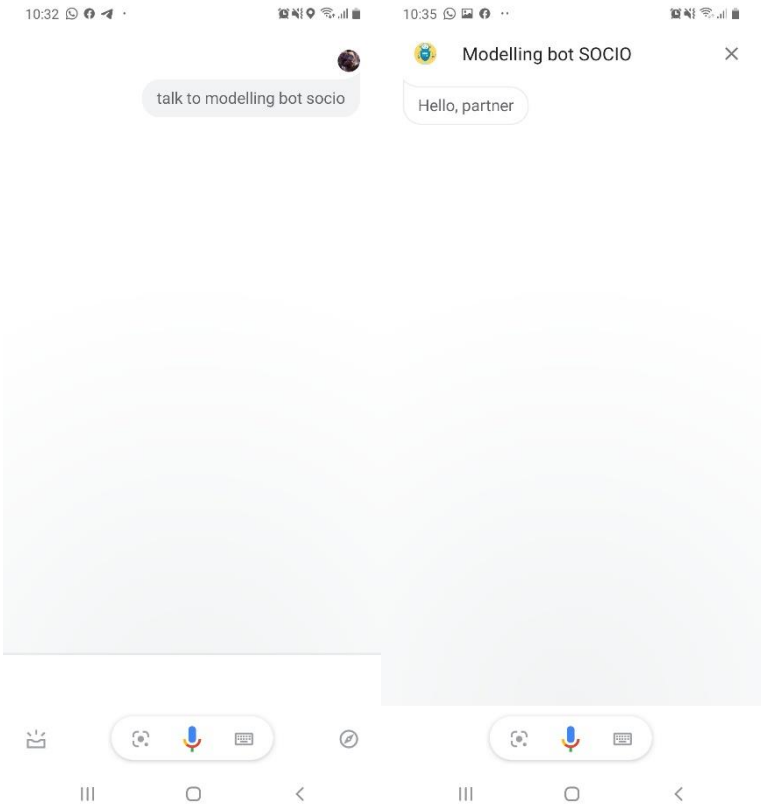


Figura 6-1: Comando de inicio del chatbot

La página web muestra el nombre del proyecto que el usuario ha entrado, junto con el último resultado, los 2 anteriores y el mensaje que hizo la acción de cada uno. En función de la anchura del dispositivo con el que se visualiza la página web, los resultados se mostrarán en vertical, en caso de que se use el dispositivo móvil en vertical, u horizontal, con el último resultado a la izquierda y los dos anteriores a la derecha.

Para la visualización de estadísticas e historiales, se mostrarán en el lugar que ocupan los 2 resultados anteriores, manteniendo el último resultado.

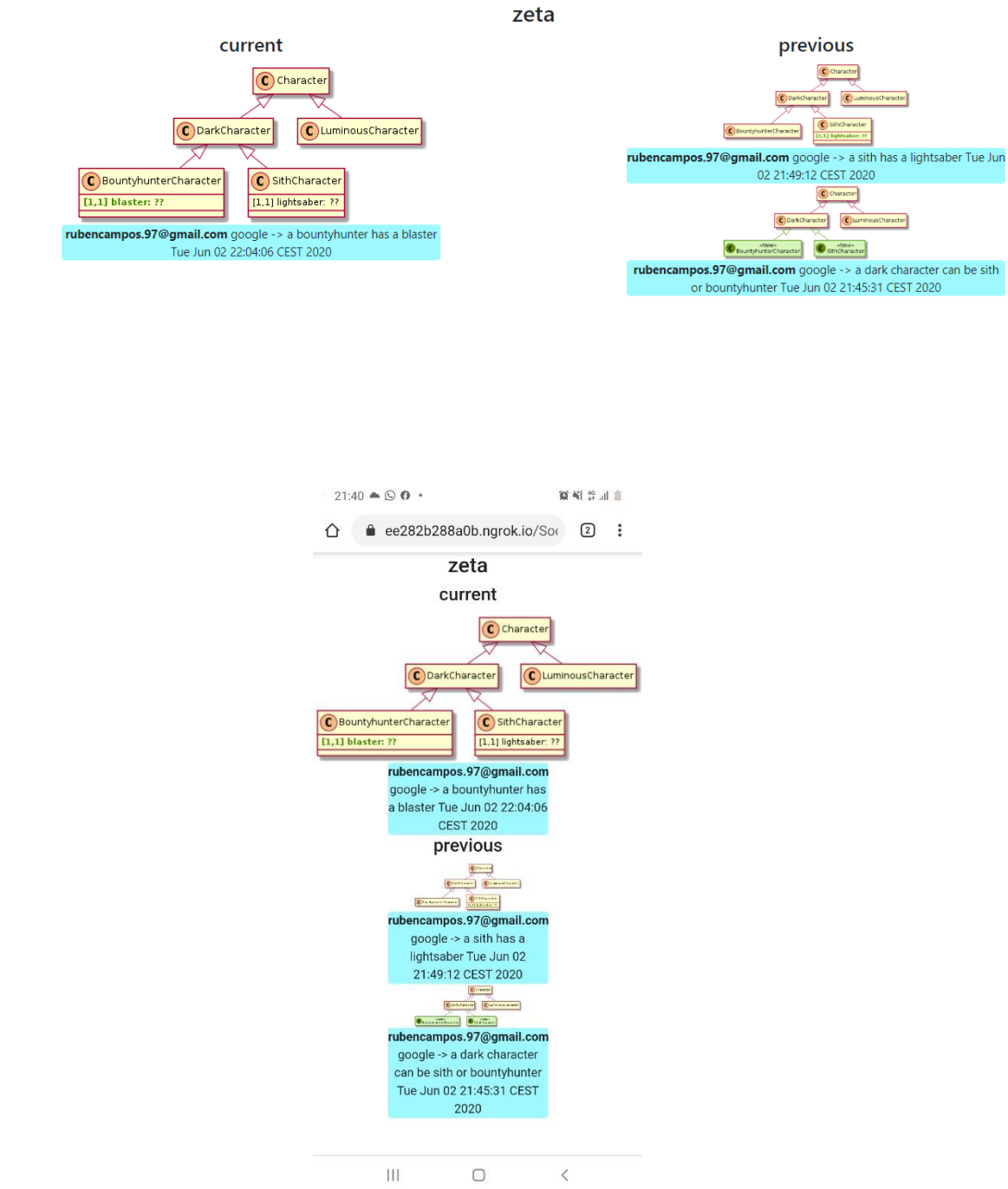


Figura 6-2: Vista de la página web en PC y en móvil respectivamente

6.2 Pruebas y resultados

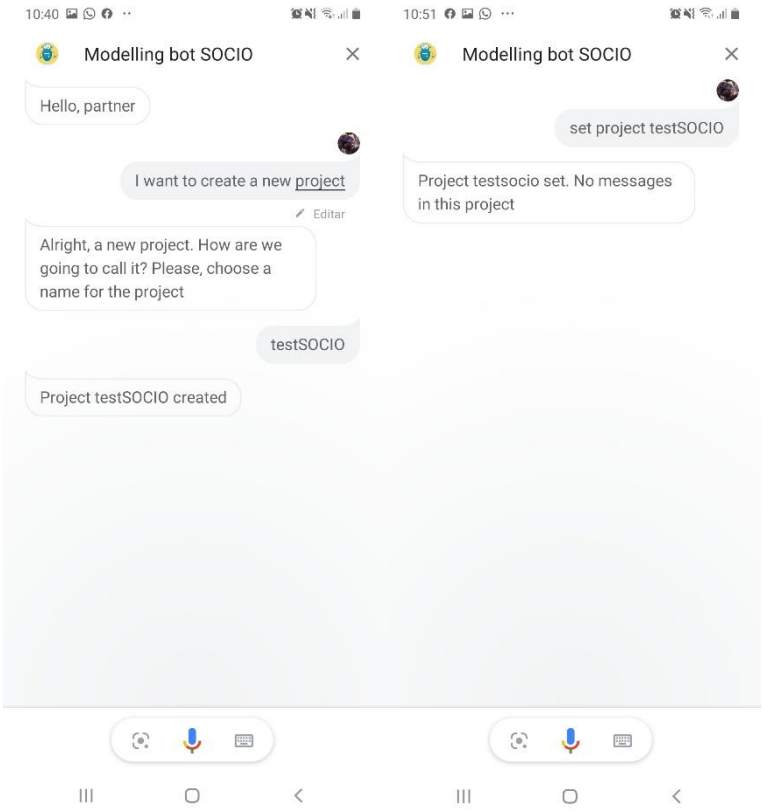


Figura 6-3: Comandos newProject y setProject

En la figura 6-3 se muestra la creación de un proyecto y acceder a uno ya creado. Si no se especifica la visibilidad, el proyecto es público por defecto. Si no se da un nombre, el bot te pedirá uno. Cuando se accede al proyecto, el bot responde si se ha realizado con éxito, y en caso positivo, se indicará si hay mensajes y si el último mensaje es del usuario en cuestión, con el objetivo de informar si ha habido últimos cambios.

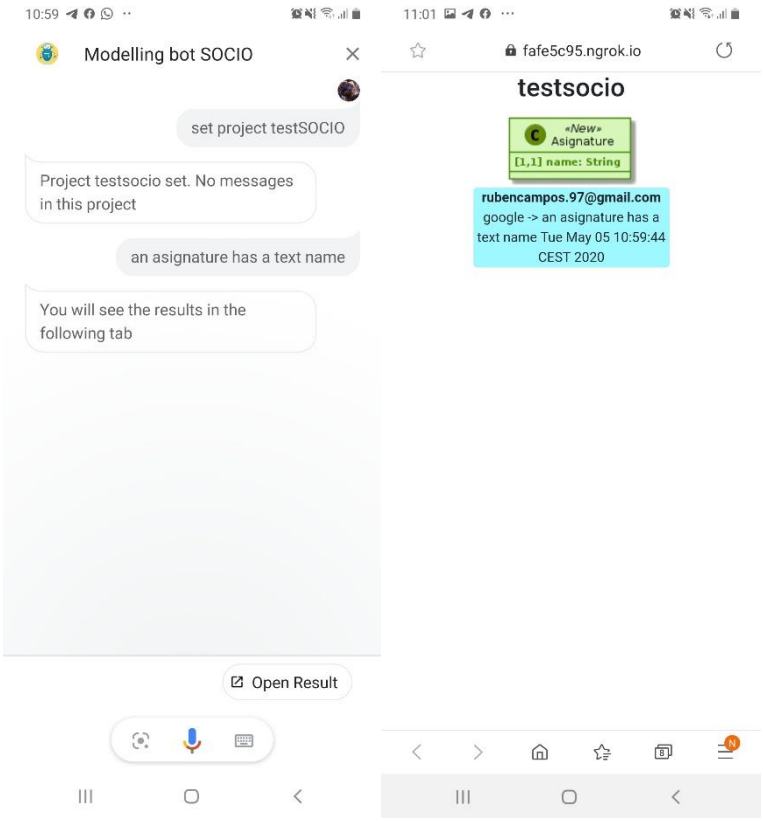


Figura 6-4: Ejemplo de frase de modelado y vista en la página web

En la figura 6-4, se muestran ejemplos cuando se envía una frase con la intención de hacer modelado. Google Assistant en versión móvil muestra el enlace de la página. Para Google Home existe una intención que deletrea la url. Se puede ver cómo se invoca pidiendo ayuda al bot sobre los proyectos (Véase Figura 6-11).

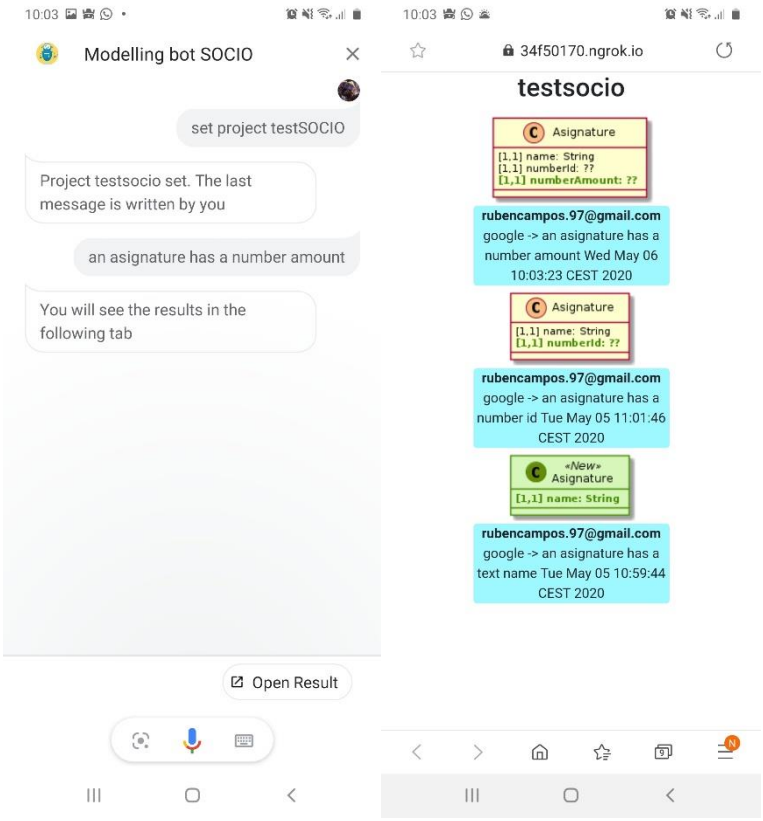


Figura 6-5: Vista de hasta 3 imágenes del diagrama

En la figura 6-5 se muestra otra frase de modelado después de otras dos acciones de modelado. Las anteriores se muestran debajo de éste en orden, permitiendo al usuario visualizarlos y poder decidir si mantener esos cambios o rehacerlos de nuevo con las acciones undo y redo que se muestran en la figura 6-6.

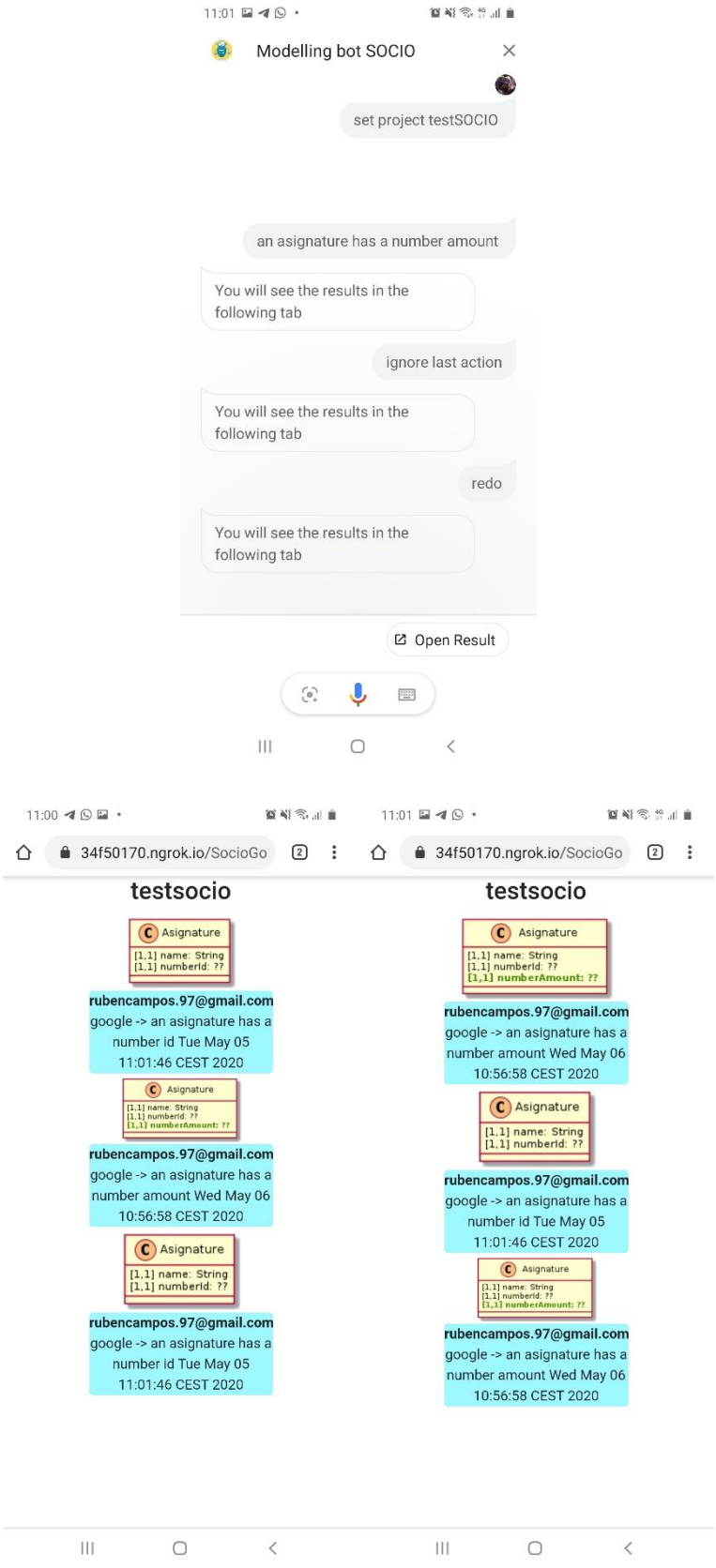
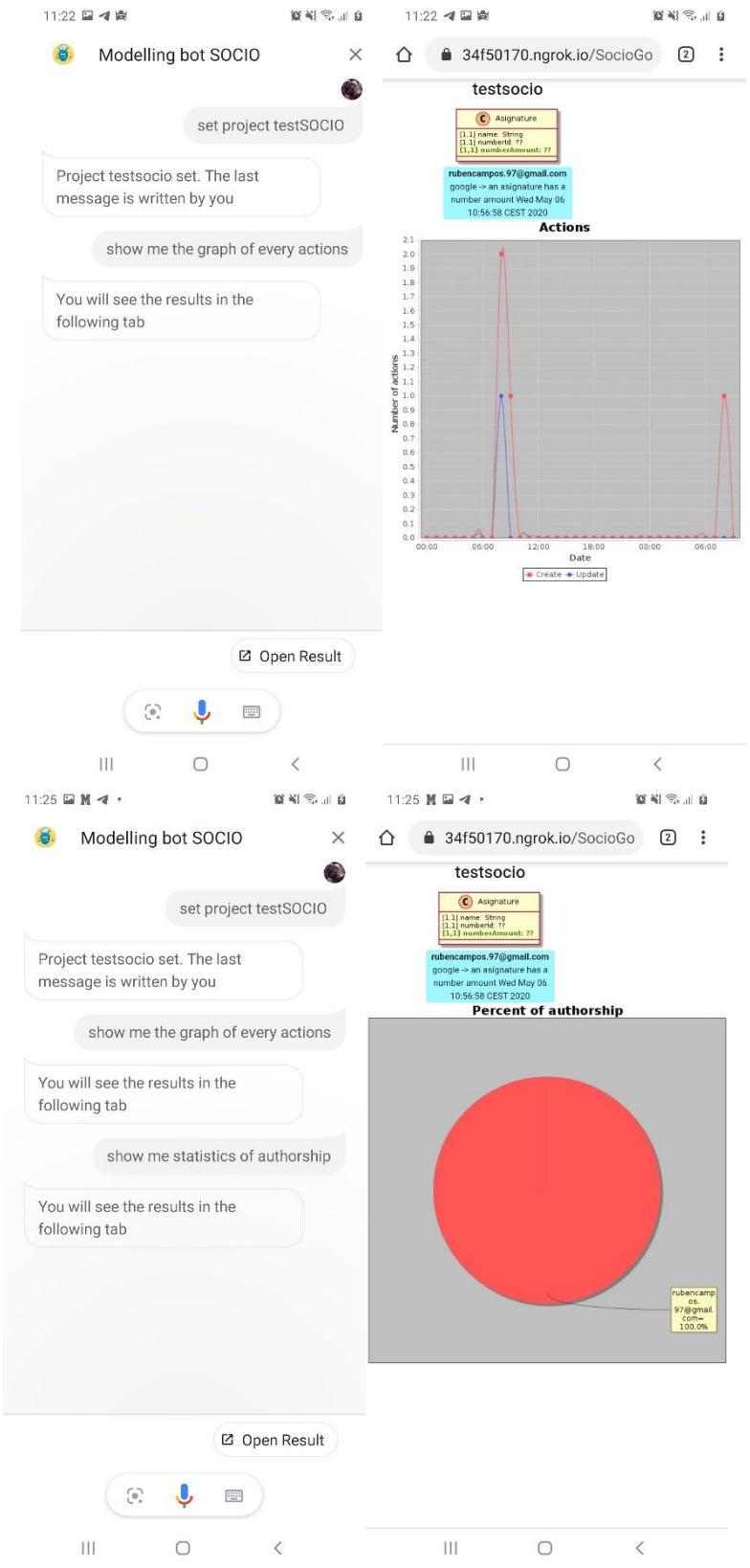


Figura 6-6: Comandos undo y redo

La figura 6-6 muestran el resultado de los comandos undo y redo respectivamente. El undo no borra la última imagen, la guarda como si fuera la acción anterior, al igual que el redo.



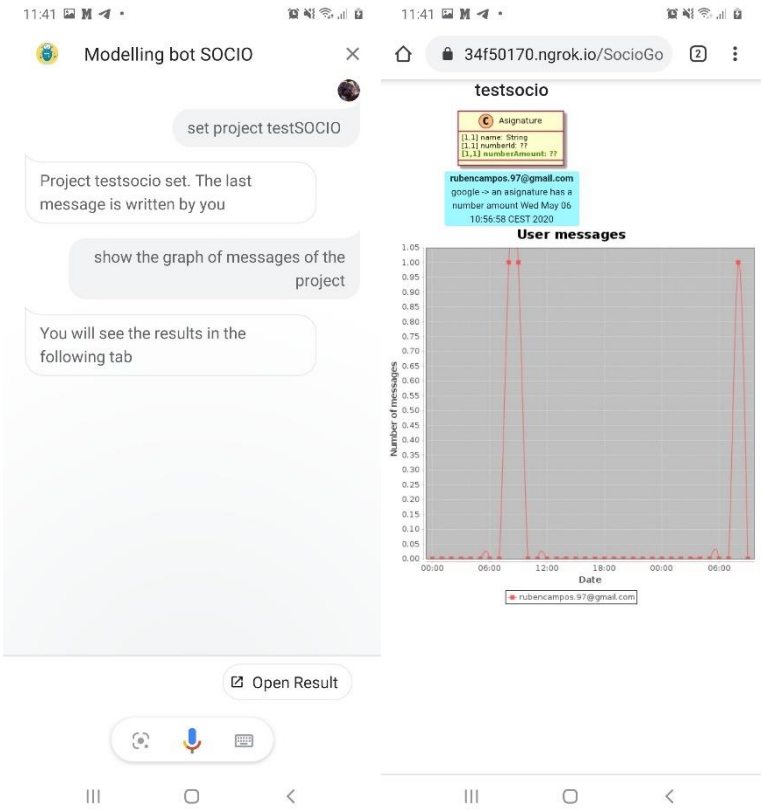
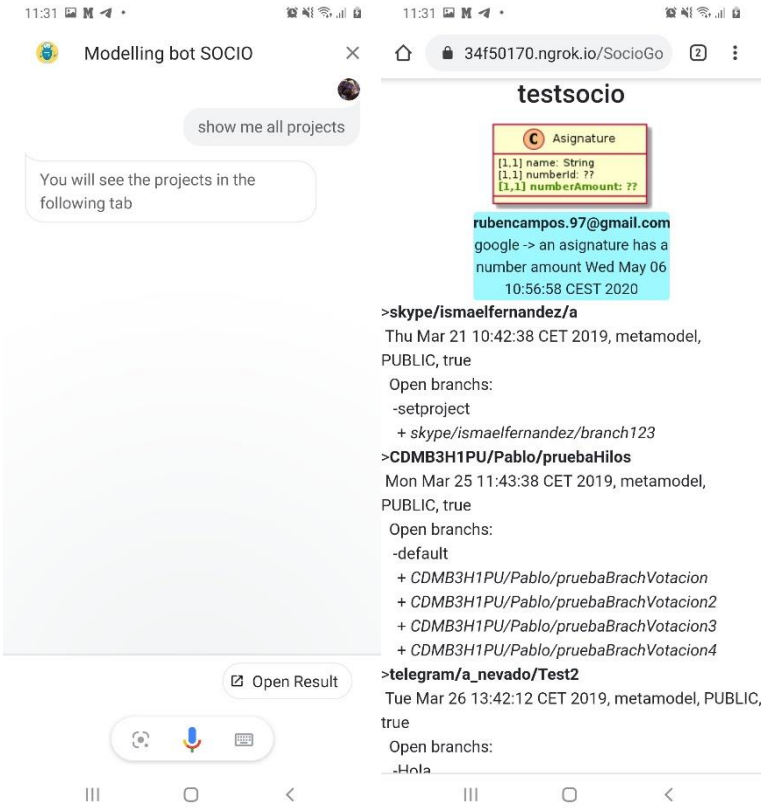


Figura 6-7: Ejemplos de estadísticas solicitadas (acciones y autoría y mensajes)

La figura 6-7 muestran las estadísticas del proyecto: acciones, autoría y mensajes por usuario. La gráfica ocupa el lugar de los 2 diagramas anteriores al último.



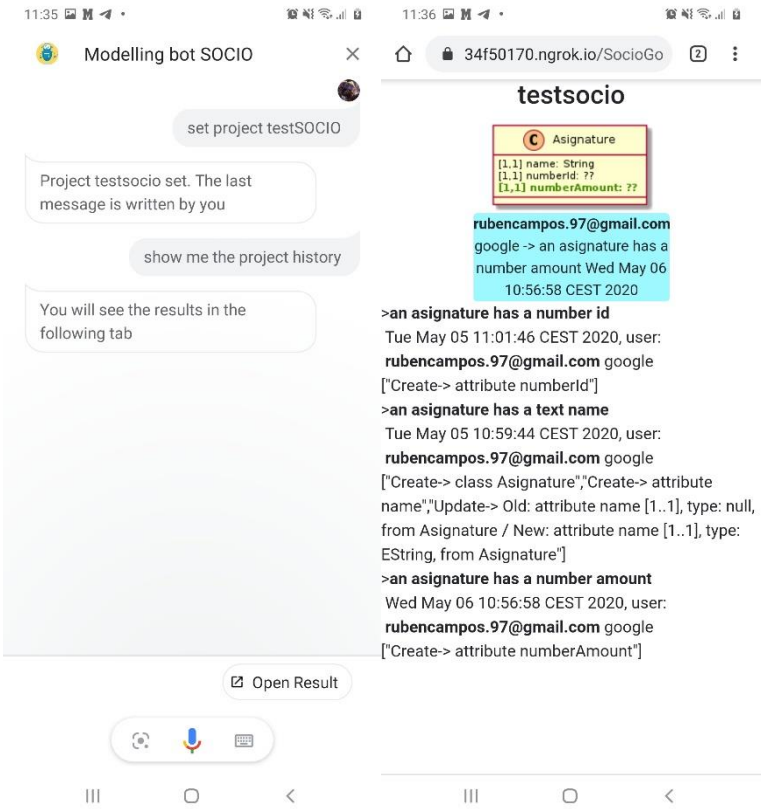
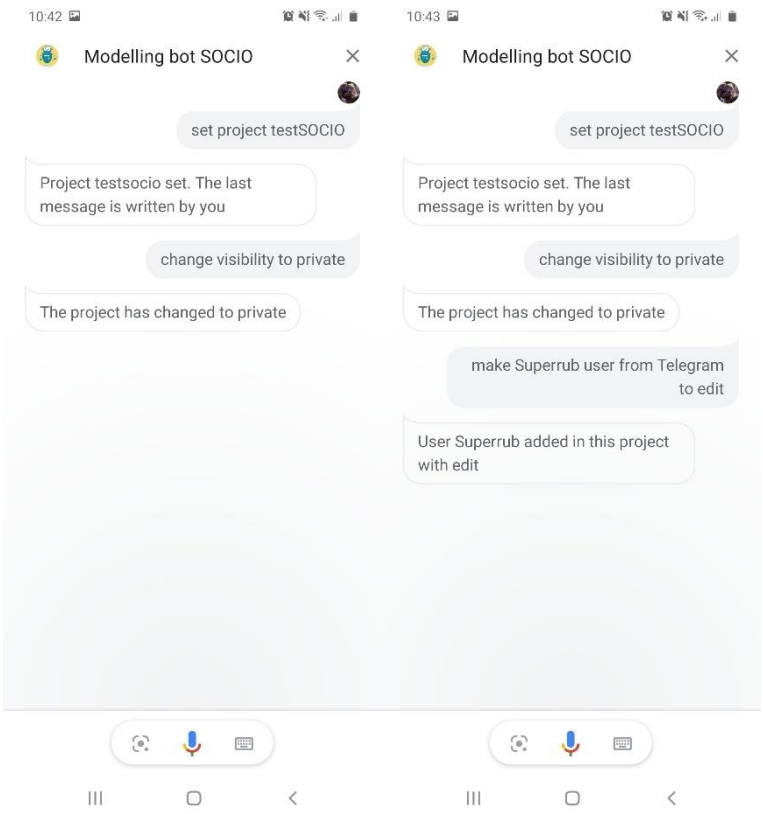


Figura 6-8: Comandos projects (listado de proyectos) e history (historial de mensajes)

En la figura 6-8 se muestra el listado de todos los proyectos y el historial de mensajes. Al igual que las gráficas, se muestran en el mismo lugar de los dos anteriores diagramas. En este caso, es texto lo que se envía y no una imagen.

Figura 6-9: Salir del bot. Cierra la ventana del bot y vuelve al inicio de Google Assistant (Figura 6-1)



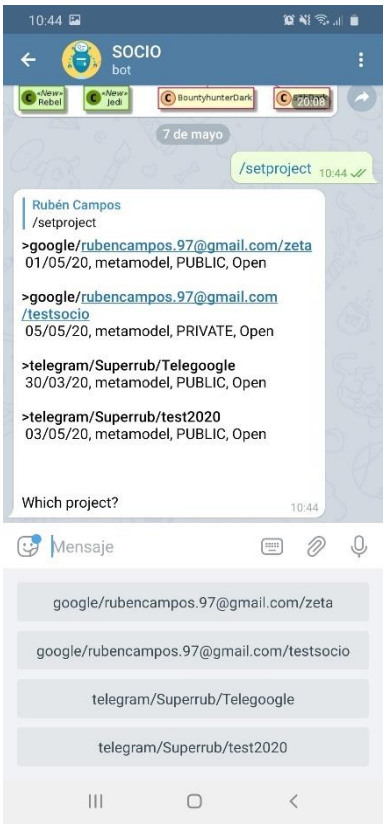


Figura 6-10: Ejemplo de añadir un usuario a un proyecto privado

En la figura 6-10, se añade a un usuario a un proyecto (que lo hacemos privado para restringir la accesibilidad). Indicamos la red social y el nombre del usuario como se muestra en la figura con el permiso que se otorga. Debido a que los usuarios son nombres propios, esta acción sólo está disponible por escrito. Si no se indica la red social, será Google por defecto.

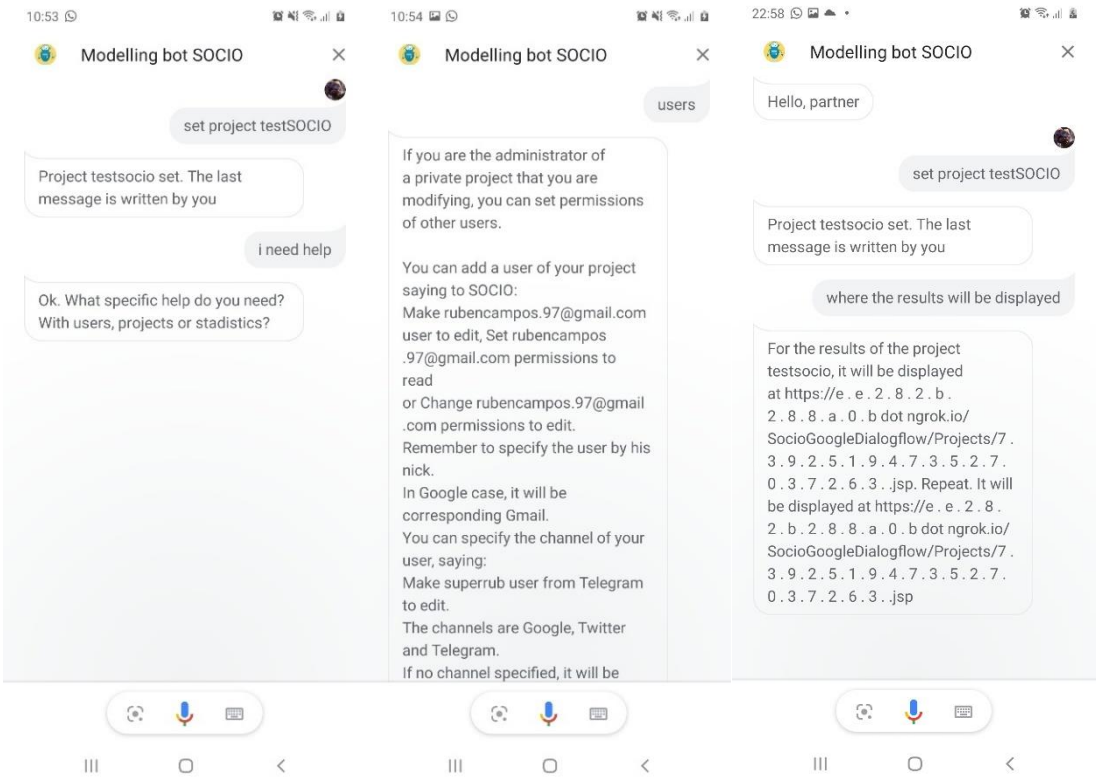


Figura 6-11: Comando help y pedir url

Se muestra una de las ayudas posibles en la figura 6-11, en este caso es de usuarios, pero por el mismo se puede ver ayuda sobre proyectos y sobre las estadísticas. También se muestra cómo pedir la url. Esta última acción está hecha para que deletree la url, de manera que, al usar un altavoz, se entienda la url. Si se usa Google Assistant en el smartphone, esta petición no es necesaria, ya que te deja acceder a ella en la parte inferior del diálogo.

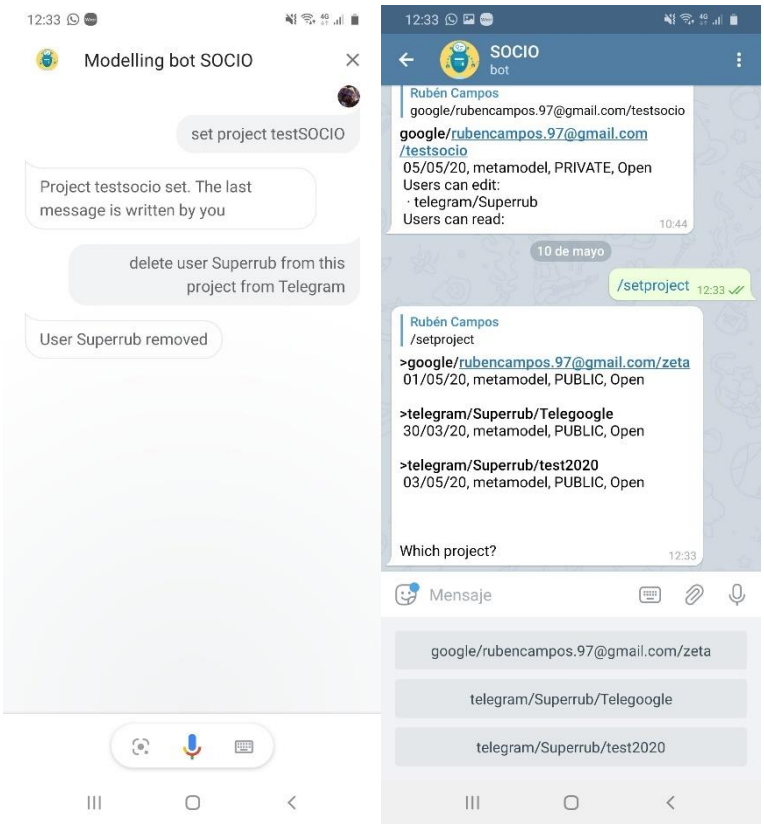


Figura 6-12: Ejemplo de quitar a un usuario de un proyecto

Al igual que en la figura 6-10, de la misma manera se puede quitar un usuario de un proyecto. No ocurrirá nada si dicho usuario no existe, aunque se responda. Al igual que en la figura 6-10, esta acción sólo está disponible por escrito y se tiene que especificar la red social.

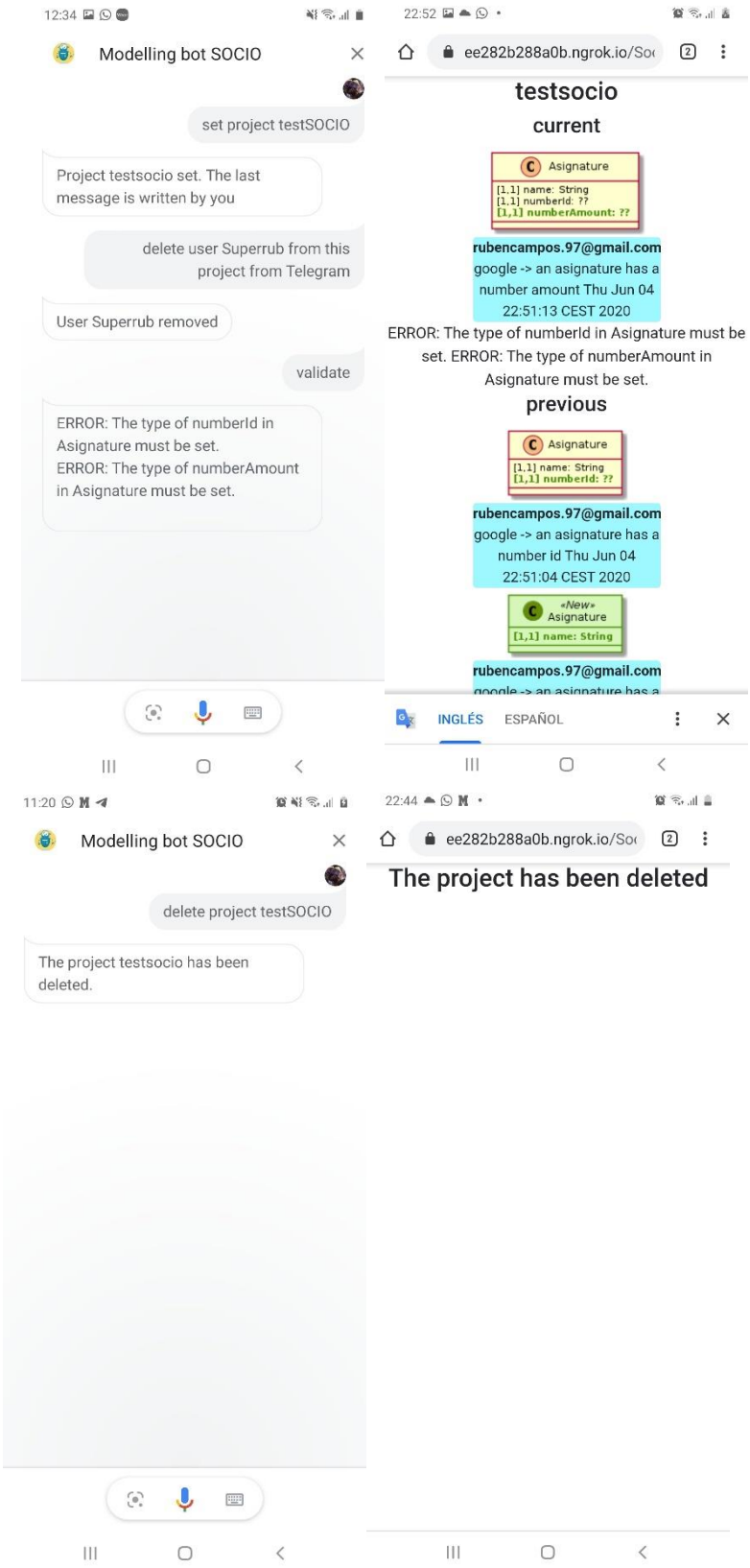


Figura 6-13: Comandos validate y removeProject (se muestra la página vacía cuando su proyecto asignado desaparece)

Esta figura refleja las dos formas de finalizar un proyecto, que sería validarlo o borrarlo. En el primer caso, devuelve el mensaje directo de SOCIO, mientras que el segundo borra el proyecto. Se puede comprobar que la página web del usuario se ha vaciado cuando se ha borrado el proyecto, indicando también un mensaje.

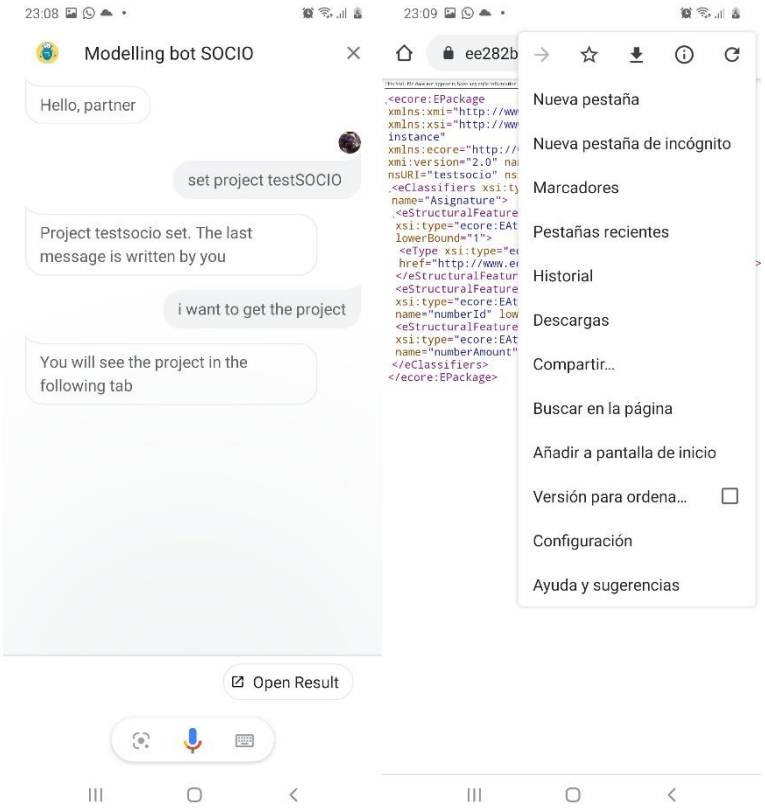


Figura 6-14: Obtener el proyecto

En esta última figura se muestra cómo se puede obtener el proyecto construido en Google Assistant. Se envía a un enlace que contiene el fichero ecore del proyecto, pudiendo acceder a él igual que en las páginas por el botón en la parte inferior. En caso de que se haga por voz, especialmente para los altavoces, el bot deletreará la url. Una vez en el enlace, se puede descargar como fichero ecore. En caso de que se guarde como xml debido a algunos navegadores, se puede eliminar la extensión .xml para mantener el formato original.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

El avance del procesamiento del lenguaje natural está aumentando según pasan los años, haciendo que cada vez existan más bots conversacionales o chatbots en varias redes sociales. Para los desarrolladores es importante tener herramientas que faciliten las tareas que componen el ciclo de vida del software, y mucho más cuando son bots los que lo facilitan, ya sea en repositorios o en redes sociales.

Por ese motivo, integramos SOCIO a Google Assistant, como objetivo de que se pueda interactuar con él mediante la voz, ya que, hasta ahora, se usa mediante mensajes escritos.

Con la ayuda de Dialogflow, nuestro backend envía la información sacada del mensaje del usuario a SOCIO y éste le responde con el resultado, mostrando el resultado en la página web del usuario.

Esta solución permite además al usuario que en una misma página web, pueda ver tanto el resultado actual como los anteriores u otros datos que pueda proporcionar SOCIO como las estadísticas de mensajes por usuario.

.

7.2 Trabajo futuro

Este trabajo soporta el funcionamiento imprescindible de SOCIO. Para un trabajo futuro, se planea introducir mecanismos que ayuden a grupos de trabajo a la toma de decisiones de un proyecto, también por voz.

También está abierto a que SOCIO pueda ser multilenguaje, ya que está en inglés, para que pueda aumentar la accesibilidad de la herramienta a más personas.

Por último, crear más herramientas chatbots, ya sean por voz o por escrito, que ayuden a los desarrolladores en sus tareas, incluso de programación, para poder hacerlas en cualquier lugar y en cualquier momento.

Referencias

[1] Lenin Medeiros, Charlotte Gerritsen, Tibor Bosse,"Towards Humanlike Chatbots Helping Users Cope with Stressful Situations", Lecture Notes in Computer Science, 9 de Agosto 2019, Vol 11683, pp.232-243

[2] Lenin Medeiros, Tibor Bosse, "Empirical Analysis of Social Support Provided via Social Media", Lecture Notes in Computer Science, 19 de Octubre 2016, Vol.10047, pp.439-453

[3] Sara Perez Soler, Esther Guerra, Juan de Lara, "Collaborative modeling and group decision making using chatbots in social networks", IEEE Software, Noviembre 2018, Vol.35(6), pp.48-54, DOI 10.1109/MS.2018.290101511

[4] "Los mejores bots para optimizar tu proceso de desarrollo", ingenieriadesoftware.es, 7 de Enero 2018 https://ingenieriadesoftware.es/bots-desarrollo-software-open-source/ [Fecha de último acceso: 23/04/2020]

[5] Angie Gonzalez, Arlene Perez, "First-timers-bot", GitHub, 2017, https://github.com/hoodiehq/first-timers-bot [Fecha de último acceso: 23/04/2020]

[6] Schachte, "Git Enforcer", GitHub, 2017, https://github.com/Schachte/Git-Enforcer [Fecha de último acceso: 01/06/2020]

[7] Huu.la "CSSRooster, a bot that writes CSS Classes for HTML with Deep Learning", 30 de Enero 2017, https://huu.la/ai/cssrooster [Fecha de último acceso: 01/06/2020]

[8] "Dependabot, Automated dependency updates", 2019, https://dependabot.com/ [Fecha de último acceso: 24/04/2020]

[9] Kévin Maschtaler. "Introducing Sedy, the Serverless GitHub Bot That Fixes Typos for you", 28 de febrero 2017, https://marmelab.com/blog/2017/02/28/sedy-the-serverless-github-bot-which-fix-typos-for-you.html#disqus_thread [Fecha de último acceso: 27/04/2020]

[10] "AWS charbot", https://aws.amazon.com/es/chatbot/ [Fecha de último acceso: 27/05/2020]

[11] "Hubot", https://sociogrupo.slack.com/apps/A0F7XDU93-hubot [Fecha de último acceso: 27/05/2020]

[12] "Unified Modeling Language". http://uml.org/ [Fecha de último acceso: 15/04/2020]

[13] "ArgoUML", https://es.wikipedia.org/wiki/ArgoUML [Fecha de último acceso: 15/05/2020]

[14] "Gliffy",https://www.gliffy.com/ [Fecha de último acceso: 07/06/2020]

[15] "Draw.io",https://drawio-app.com/ [Fecha de último acceso: 07/06/2020]

[16] "Cacoo by Nulab",https://cacoo.com/es/ [Fecha de último acceso: 15/05/2020]

[17] "Cawemo",https://cawemo.com/ [Fecha de último acceso: 15/05/2020]

[18] "Object Management Group business Process Model and Notation", http://www.bpmn.org/ [Fecha de último acceso: 06/05/2020]

[19] "SOCIO Modelling Bot", Sara Pérez Soler Github, 2018 https://saraperezsoler.github.io/ModellingBot/ [Fecha de último acceso: 22/05/2020]

[20] Sandy Mccarthy"Alexa vs Google Assistant vs Siri: Google's Voice Assistant has higher IQ", University Wire, 21 de Agosto 2019

Glosario

API	Application Programming Interface